ITRI 615

Pfleeger Chapter 5 Part 1

Whitman & Mattord Chapter 7

Protection in operating systems

Separation is the basis of protection in operating systems. Keeping one user's objects separate from other users.

- Physical separation. Where different physical objects are used, for example printers.
- Temporal separation. Processes with different security requirements are executed at different times.
- Logical separation. Where users operate under the illusion that no other processes exist.
- Cryptographic separation. Where processes conceal their data and computations from outside processes.

Protection in operating systems *cont.*

- An operating system can support separation and sharing in several ways, offering protection at any of several levels.

1. Do not protect.
2. Isolate.
3. Share all or share nothing.
4. Share via access limitation.
5. Share by capabilities.
6. Limit use of an object.

Memory and address protection

- The most obvious problem of multiprogramming is preventing one program from affecting the data and programs in the memory space of other users. Fortunately many protection mechanisms exist which include:

1. Fences. A method to confine users to one side of a boundary (see figure 5.6, p. 298).

2. Relocation. Taking a program written as if it began at address 0 and changing all addresses to reflect the actual address at which the program is located in memory.

Memory and address protection *cont.*

3. Base/bounds registers. Base (or fence) registers provide a starting address where programs will be loaded for execution, but no upper one. Bounds registers are upper address limits.

4. Tagged architecture. Every word of machine memory has one or more extra bits to identify the access rights to that word. These access bits can be set only by privileged operating system instructions.

5. Segmentation. Involves the simple notion of dividing a program into separate pieces.

6. Paging. Program is divided into equal-sized pieces called pages, and memory is divided into equal-sized units called page frames. Paging and segmentation can also be combined.

Control of access to general objects

- Protecting objects is a more general problem than protecting memory.

- There are several complementary goals in protecting objects, namely:

1. Check every access. Every access by a user to an object should be checked.

2. Enforce least privilege. A subject should have access to the smallest number of objects necessary to perform some task.

3. Verify acceptable usage. After access has been granted, the intended action to be performed should also be verified.

Control of access to general objects *cont.*

- Several object access control mechanisms exist, including:

1. Directories. A file directory that has a unique owner who is able to grant and revoke access rights. This directory lists all the files to which that user has access.

2. Access control list. One list for each object, indicating who has access to it and what access they have.

3. Access control matrix. A table in which each row represents a subject, each column represents an object, and each entry is the set of access rights for that subject to that object.

4. Capability. A capability is a ticket giving permission to a subject to have a certain type of access to an object.

Control of access to general objects *cont.*

5. Kerberos. Implements both authentication and access authorization by means of capabilities, or tickets, secured with symmetric encryption. Requires two systems, the authentication server and the ticket-granting server.

6. Procedure-oriented access control. A procedure that controls access to objects, in addition to the protection provided by the operating system.

7. Role-based access control. Lets us associate privileges with groups, i.e. administrators vs. standard users.

File protection mechanisms

As the number of users increase, so too does the complexity of file protection schemes.

1.All-none protection. In early operating systems all files were public. Users could perform any action on anybody's file. The only "protection" involved trust along with ignorance. It was felt that you would treat the files of other users as you would like yours to be treated. Only option was passwords, if needed.

It was unacceptable for several reasons.

•Lack of trust.

•Too coarse.

•Rise of sharing.

•Complexity.

•File listings.

File protection mechanisms *cont.*

2. Group protection. Focused on identifying groups of users who had some common relationship. Group members then had access to the files of only the other members of the group. It therefore overcame some of the shortcomings of all-none protection, but caused some new difficulties.

- Group affiliation.
- Multiple personalities.
- All groups.
- Limited sharing.

File protection mechanisms *cont.*

Individual permissions was the next attempt to improve file protection.

3. Persistent permission. A mechanism which uses access lists and tokens or passwords to control access. Revocation of access is relatively difficult.

4. Temporary acquired permission. Based on a three-level user-group-world hierarchy. Utilizes a permission called **set userid** (suid). If this protection is set for a file to be executed, the protection level is that of the file's owner, not the executor. For example, suppose Tom owns a file and allows Ann to execute it with suid. When Ann executes the file, she has the protection rights of Tom, not of herself.

User authentication

- An operating system bases much of its protection on knowing who a user of the system is.
- Authentication mechanisms use any of three qualities to confirm a user's identity. These qualities are:

1. Something the user knows.
2. Something the user has.
3. Something the user is.

User authentication *cont.*

- Passwords are mutually agreed-upon code words, assumed to be known only to the user and the system. They are the most common authentication mechanism for user to operating system.

- They do however have some difficulties.

1. Loss.
2. Use.
3. Disclosure.
4. Revocation.

- Additional authentication information can also be used, for example office hours, IP addresses, etc. This is referred to as multifactor authentication.

User authentication *cont.*

- Attacks on passwords can take many forms. Listed below, in decreasing order of difficulty, are ways which might be tried to obtain a user's password.

1. Try all possible passwords.
2. Try frequently used passwords.
3. Try passwords likely for the user.
4. Search for the system list of passwords.
5. Ask the user.

- Loose-lipped systems. More often than not password authentication systems are their own enemies. For example, entering an invalid user name usually leads to an "invalid user name" response, similarly an incorrect password.

User authentication *cont.*

- Exhaustive (brute force) attack. The attacker tries all possible passwords, usually in some automated fashion. The number of possibilities depends on the particular computing system.
- Probable passwords. Think of a word. Is it long, uncommon, or hard to pronounce or spell? Penetrators looking for passwords exploit these human traits.
- Passwords likely for a user. We tend to choose our passwords according to something that is meaningful to us.

User authentication *cont.*

- Plaintext system password list. To validate passwords, the system must compare them with actual passwords. On some systems the password list is a file, containing two columns with user ID's and corresponding passwords. Merely protecting this table is often insufficient.

- Encrypted password file. Similar to the plaintext list, with the only difference being the fact that the ID's, Passwords, or both are encrypted. When a password is entered, the comparative data is decrypted, and then compared to check for validity.

- Indiscreet users. As users tend to forget passwords or might be just plain lazy, they tend to keep record of their passwords on or around their computers. Research has shown that users sometimes even tend to simply divulge their passwords, for no obvious reason.

User authentication *cont.*

- Several guidelines have been developed for password selection.

1. Use characters other than just A – Z.
2. Choose long passwords.
3. Avoid actual names or words.
4. Choose an unlikely password.
5. Change the password regularly.
6. Don't write it down.
7. Don't tell anyone else.

User authentication *cont.*

- The current trend in user authentication is moving towards biometrics. Biometrics are biological authenticators based on some physical characteristic of the human body. Current technologies include fingerprints, hand geometry, retina and iris, voice, handwriting, blood vessels in the finger and face.

- There are however several problems with biometrics.

1. Relatively new, and considered by some to be intrusive.
2. Recognition devices are costly.
3. Readers use sampling to decide on being close enough.
4. Can become a single point of failure.
5. There are still false readings.
6. The speed required limits accuracy.
7. Although we think biometrics are unique, forgeries are possible.

Questions

- Any questions?